Cours Analyse Multi-Résolution

Imagerie Satellite et Compression

E. Christophe
emmanuel.christophe@tesa.prd.fr

24 mai 2005

Table des matières

L'observation de la Terre depuis des satellites a donné naissance à un grand nombre d'applications depuis de nombreuses années. Applications météorologiques, cartographie, renseignement ou applications scientifiques, le domaine est très vaste.

Les premiers satellites utilisaient la technique classique de photographie, posant le problème délicat de la récupération des films. La mise au point de capteurs imageurs numériques (Charge Coupled Device ou CCD), dans les années 70, a permis la transmission de l'image par liaison radio. Ces technologies sont à l'origine de l'essort de la télédétection par satellite.

La numérisation de l'information rend possible des traitements destinés à faciliter la transmission des données. Parmi ces traitements, la compression est devenue une étape indispensable permettant d'accroître les capacités d'acquisition. Depuis l'apparition de la compression dans la chaîne d'acquisition des images satellites, de grands progrès ont été faits, rendant possible une meilleure précision des données acquises. Les innovations introduites dans les satellites suivent de près les techniques adoptées dans les traitements au sol.

FIG. 1 – L'INSA Toulouse vu par SPOT-5

1 La compression et l'imagerie spatiale

1.1 Les images

Les images satellites sont de même nature et possèdent des propriétés similaires à la plupart des images photographiques (Fig. 1). Les évaluations de systèmes de com-

pression peuvent donc se faire sur des images ordinaires. En général, il est préférable de travailler sur des images standards comme *Lena* ou *Barbara* pour faciliter les comparaisons entre différents travaux.

Une image, vue de manière numérique, est constituée d'un ensemble de pixels organisés sur une grille carrée (Fig. 2). Chacun des pixels (picture element) prend une valeur numérique. Dans le cas des images en niveau de gris, cette valeur est souvent comprise entre 0 (noir) et 255 (blanc) comme on le voit sur Fig. 3.

On peut noter l'image comme une fonction discrète à deux paramètres : $u(m, n)$. Chacune des valeurs de $u(m, n)$ correspond à la valeur du pixel situé sur la colonne m et la ligne n.

FIG. 2 – *Lena* et un détail de l'image

112	77	56	66	157	204	204	198
108	74	53	87	177	207	204	204
72	57	66	126	197	209	206	208
55	65	113	173	207	211	210	211
77	107	160	198	208	206	208	207
127	158	188	202	203	202	202	206
163	186	197	198	202	205	207	209
182	187	193	197	199	203	206	209

FIG. 3 – Valeurs du détail de l'image *Lena* (Fig. 2)

L'intervalle possible des valeurs est lié à la taille de la représentation binaire accordée à chaque pixel. Le cas le plus courant est celui des images 8 bits. Cela signifie que chaque pixel sera codé sur 8 bits ce qui permet $2^8 = 256$ valeurs différentes.

Certaines applications qui ont besoin de plus de précision pour la représentation de ces niveaux utiliseront des représentations sur 12 ou 16 bits.

Le cas des images couleurs est un peu particulier. Il s'agit en fait d'une combinaison de 3 images en niveau de gris, représentant respectivement le rouge, le vert et le bleu. Pour les satellites, un principe similaire est utilisé, combinant 3 capteurs pour produire les images rouge, verte et bleue. Souvent, un capteur infrarouge est également utilisé car il apporte des informations importantes sur la végétation. On parle dans ce cas d'images multispectrales.

Les images sont très proches du traitement du signal. Une ligne de l'image peut être vue comme un signal (Fig. 4). Par analogie, on définit la notion de fréquence spatiale

liée à des phénomènes périodiques sur l'image. C'est ce qu'on peut voir lorsqu'on effectue la transformée de Fourier d'une image par exemple.

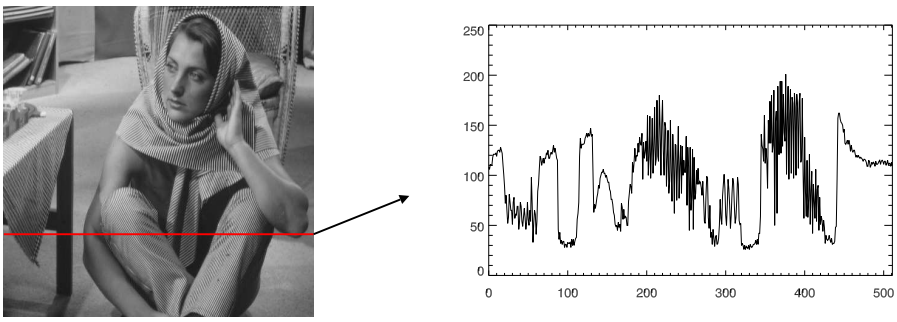


FIG. 4 – Profil d'une ligne de *Barbara*. On voit clairement l'apparition de fréquences spatiales.

1.2 Pourquoi la compression ?

1.2.1 Taille des données

La taille des données pour une image (en bits) est données par $N_{lignes} * N_{colonnes} * B$ où N_{lignes} est le nombre de pixels par ligne, $N_{colonnes}$ le nombre de pixels par colonne, et B le nombre de bits par pixel (classiquement 8 bits pour une image en niveau de gris).

Pour avoir une image, le nombre de pixel nécessaire est important. Par exemple, l'imagette de la Fig. 2 contient déjà $512 * 512 = 262\,144$ pixels. Les images satellites demandent souvent à couvrir une large zone avec une très bonne résolution. Des images contenant 500 millions de pixels sont donc courantes.

L'évolution des techniques d'acquisition tend à accroître la taille des données comme on peut le voir dans Tab. 1.

	SPOT 1-3	SPOT 5	Pléiades
Année	1986-1993	2002	fin 2008
Instrument	HRV	HRG	HR
Fauchée	60 km	60 km	20 km
Résolution	10 m	5 m	0.7 m
Codage	8 bits	8 bits	12 bits
Taille scène	36 Mo	144 Mo	$\sim$1000 Mo
Taux Comp.	1.3	2.8	Ajustable
			$\sim$5

TAB. 1 – Évolution de la taille des données pour différents satellites d'observation du CNES.

1.2.2 Mesures utilisées

Différents termes techniques sont utilisés lorsque l'on parle de compression. Le débit indique le nombre moyen de bits utilisés pour coder un pixel. Le débit après

compression peut être mesuré en divisant la taille de l'image compressée par le nombre de pixels. Ce débit est mesuré en bits par pixel (noté bpp)

Le rapport entre le nombre de bits par pixel avant et après compression est appelé taux de compression. Par exemple, pour une image codée au départ sur 8 bits, compressée avec un débit de 2 bpp, on parlera d'un taux de compression de 4 :1 (1 bit dans l'image compressée pour 4 bits de l'image originale en moyenne).

Dans le cas où l'image obtenue après décompression n'est pas exactement semblable à l'image originale (cas de la compression avec pertes), il est nécessaire de mesurer la différence pour juger de son importance. On définit des mesures de distortion (ou critère de qualité). Le plus courant est l'Erreur Quadratique Moyenne (EQM), ou Mean Square Error (MSE) en anglais. En notant $\tilde{u}(m,n)$ l'image reconstruite, n_{lin} et n_{col} respectivement le nombre de lignes et de colonnes de l'image, on a :

$$EQM = MSE = \frac{1}{n_{lin} n_{col}} \sum_{m,n} \left(u(m,n) - \tilde{u}(m,n)\right)^2 \tag{1}$$

Une autre mesure très utilisée est le PSNR (Peak Signal to Noise Ratio) définie par :

$$PSNR_{(dB)} = 10. \log_{10} \frac{Peak\ Signal^2}{MSE} \tag{2}$$

où $Peak\ Signal$ correspond à la valeur maximale qu'il est possible de coder en utilisant la dynamique de l'image.

Le MSE et le PSNR ne sont pas représentatifs de l'impression de qualité qu'on a en regardant une image : deux images ayant un MSE semblable peuvent avoir une qualité perçue très différente. D'autres mesures de distortion existent, certaines essaient de rendre compte d'une manière plus réaliste de la perception humaine. Malgré leurs défauts, le MSE et le PSNR restent très employés car ils sont simples à calculer et facilement modélisables mathématiquement.

1.2.3 Limitations

La taille très importante de ces données pose des problèmes particulièrement en spatial où les contraintes sont très fortes. Les composants électroniques embarqués à bord des satellites doivent être qualifiés avant d'être utilisés. Cette qualification est nécessaires à cause des conditions spéciales qui règnent dans l'espace, notamment les rayonnements. Les composants utilisés dans les satellites ont donc généralement quelques générations de retard comparé à ce qui se fait sur Terre.

Ce retard provoque donc une limitation dans la puissance de calcul disponible, les contraintes en terme de complexité des algorithmes de compression sont donc très fortes.

Les contraintes mécaniques empêchent bien sûr l'utilisation de systèmes tels que les disques durs dans l'espace, les mémoires utilisées sont donc des mémoires de masse dont la capacité est plus limitée. Cette limitation dans le stockage à bord est critique pour les satellites d'observation qui doivent être en visibilité d'une station de réception pour transmettre les données acquises.

Enfin, il existe une limitation à la capacité de transmission car l'environnement est très bruité. L'ordre de grandeur du débit est d'environ 100 Mbits/s.

1.3 Défi

La conception d'algorithmes de compression pour les satellites pose plusieurs défis. Tout d'abord, une fois envoyés, les algorithmes ne sont pas modifiables, il n'est évidemment pas question de ramener le satellite pour faire une mise à jour des programmes. Une phase de simulation importante a donc lieu pour s'assurer que l'algorithme fonctionnera dans la plupart des situations rencontrées.

Les contraintes sur les composants imposent des limitations importantes et il faut s'assurer tout au long du développement de l'algorithme que la solution proposée sera réalisable.

Un autre défi est posé par le problème de la régulation de débit. Les capacités de transmission du satellite au sol ou de traitement des équipements embarqués sont constantes au cours du temps. Il faut donc que le débit en bits soit le plus régulier possible en sortant de la compression. Certaines zones de l'image sont plus faciles à compresser que d'autres et donc donneront un débit plus faible. Typiquement, une zone de nuage ou de mer peut être fortement compressée sans causer une distortion trop importante. Au contraire, une zone urbaine sera plus difficile à compresser. Lorsqu'une image contient des zones mixtes, il est intéressant d'accorder plus de bits pour la zone urbaine (pour diminuer la distortion) qu'on pourra rattraper sur les zones de mer. Ces problèmes sont délicats et font toujours l'objet de recherches actives.

1.4 Avec ou sans pertes ?

Deux grandes familles d'algorithmes de compression existent. Ceux qui peuvent reconstituer l'information exacte (algorithmes sans pertes) et ceux qui tolèrent une perte d'information (algorithmes avec pertes).

Évidemment, les algorithmes sans pertes paraissent préférables à première vue. En général, les images naturelles (par opposition aux images générées par logiciel) sont complexes. Les taux de compression obtenus avec des algorithmes sans pertes sur de telles images dépassent rarement 4 ou 5. Les algorithmes sans pertes présentent donc des limitations assez importantes en terme de compression. Tolérer quelques pertes permet de dépasser ces limites. Un autre avantage des algorithmes avec pertes est qu'ils permettent en général d'ajuster le débit. Si on tolère des distortions assez importantes, on pourra compresser plus (débit plus faible) et inversement.

Par exemple les images disponibles sur un ordinateur peuvent être en différents formats. Les images .jpg sont codées avec pertes : une distortion est tolérée, tandis que les images .png sont sans pertes, l'image décompressée sera exactement identique à l'originale.

2 Compression sans perte

2.1 Source

On considère S une source d'information. Cette source est composée d'un ensemble fini de N symboles $(s_0, s_1, \ldots, s_{N-1})$ appelé alphabet, associée à un processus d'émission suivant une loi de probabilité $(p(s_0), p(s_2), \ldots, p(s_{N-1}))$, où $\sum p(s_i) = 1$.

Un message est défini par une suite de symboles $S_n = s_{i_0} s_{i_1} \ldots s_{i_{N-1}}$.

Une source est dite *sans mémoire* si les symboles sont indépendants et identiquement distribués (iid), c'est à dire $p(S_n) = p(s_{i_0})p(s_{i_1}) \ldots p(s_{i_{N-1}})$.

Une image peut être considérée comme un message contenant N symboles (N étant le nombre de pixels), chaque symbole prenant sa valeur dans un alphabet composé des niveaux de gris de 0 à 255. L'hypothèse de source iid n'est pas strictement valable pour les images naturelles.

2.2 Notion d'entropie

Dans un message, on comprend intuitivement que tous les symboles n'apportent pas la même information. Un symbole inattendu (faible probabilité) apportera plus d'information qu'un symbole attendu (probabilité forte).

Cette notion intuitive d'information a été formulée par Shannon en 1948 [1]. L'information $\mathcal{I}(s_i)$ associée au symbole s_i est fonction de la probabilité d'apparition de s_i : $p(s_i)$. Cette fonction $F(p(s_i))$ est définie par trois conditions :

1. si la source ne délivre qu'un seul message, l'information associée à ce message est nulle ;
2. en considérant s_i l'union de deux événements indépendants (s_j et s_k), $s_i = s_j \cup s_k$, avec $p(s_i) = p(s_j)p(s_k)$. L'information $\mathcal{I}(s_i)$ est égale à la somme des informations associées à s_j et s_k : $F(p(s_i)) = F(p(s_j)) + F(p(s_k))$;
3. F est continue, monotone et positive.

La fonction F vérifiant ces trois conditions est $-\lambda \log(.)$. Le coefficient λ est choisi pour que la quantité d'information associée aux n symboles d'une source soit égale à 1 lorsque les symboles sont équiprobables : $-\lambda \log(1/n) = 1$. L'unité exprimant la quantité d'information dépend de la base du logarithme. Dans le cas où $n = 2$ l'unité est le *bit*, et on a :

$$\mathcal{I}(s_i) = -\log_2(p(s_i)) \tag{3}$$

L'entropie $H(S)$ d'une source sans mémoire S suivant la loi de probabilité $p(S_n) = p(s_{i_0})p(s_{i_1}) \dots p(s_{i_{N-1}})$ est définie par :

$$H(S) = -\sum_{i=0}^{N-1} p(s_i) \log_2(p(s_i)). \tag{4}$$

L'entropie est une mesure de l'information moyenne de chaque symbole de la source. Elle est maximale si tous les symboles $S_n = s_{i_0} s_{i_1} \dots s_{i_{N-1}}$ de la source sont équiprobables.

2.3 Principe des codes à longueur variable

Le principe du codage à longueur variable (*Variable Lenght Coding* ou VLC) est d'attribuer une longueur plus importante aux symboles qui contiennent plus d'information (symboles les moins fréquents). Inversement, on attribue une longueur plus courte pour les symboles contenant moins d'information (les symboles les plus fréquents).

Par exemple dans la langue française, la lettre *e* est beaucoup plus fréquente que la lettre *z*, on va donc affecter un code plus court pour le *e* que pour le *z*. C'est ce qu'on retrouve dans le code Morse par exemple où le *e* est codé par "." tandis que le *z* est codé par "$--$..". On comprend intuitivement que la longueur totale du message sera raccourcie.

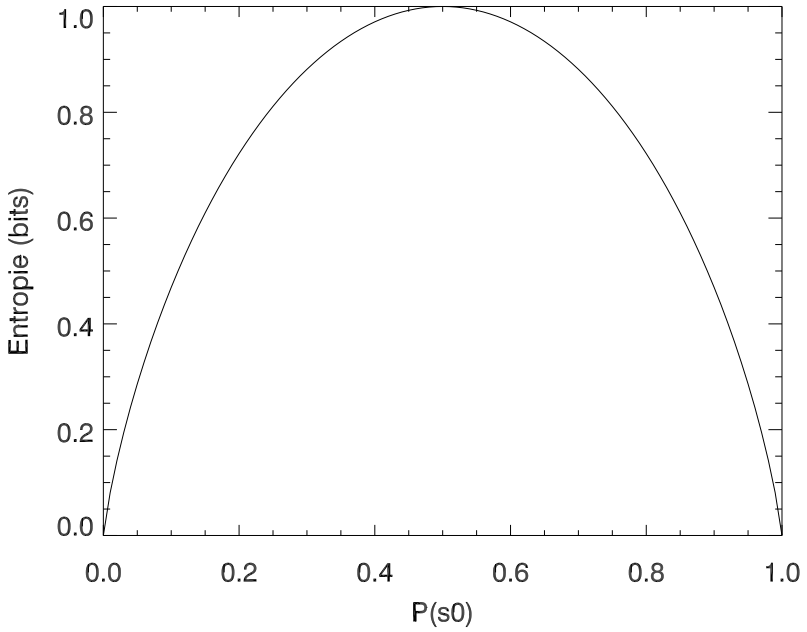


FIG. 5 – Entropie en fonction de la probabilité $p(s_0)$ pour une source à deux états (s_0, s_1) d'où $p(s_1) = 1 - p(s_0)$. L'entropie est maximale pour $p(s_0) = p(s_1) = 0.5$

On note $p(s_i)$ la probabilité d'apparition du symbole s_i et $l(s_i)$ la longueur du code affecté à s_i. On défini la longueur moyenne d'un code à longueur variable par

$$l_{moyenne} = \sum_{i=0}^{N-1} p(s_i) l(s_i) \tag{5}$$

Cette longueur moyenne ne peut pas être inférieure à l'entropie. L'entropie est donc une borne inférieure pour les codes à longueur variable.

2.4 Code de Huffman

Un codage VLC souvent utilisé est le codage de Huffman. Le but du code de Huffman est d'affecter un codage plus court pour les symboles les plus fréquents. On considère par exemple le message constitué des symboles $(s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7)$ codés en binaire sur 3 bits (cf. Tab. 2).

L'algorithme de Huffman fonctionne de la manière suivante :

1. Arranger les probabilités des symboles $P(s_i)$ par ordre décroissant et les considérer comme les feuilles d'un arbre.
2. Tant qu'il reste plus d'un nœud :
   – Regrouper les deux nœuds avec la plus faible probabilité pour former un nouveau nœud ayant pour probabilité la somme des deux nœuds regroupés.

Symbole	Code initial	Probabilité
s_0	000	0.25
s_1	001	0.21
s_2	010	0.15
s_3	011	0.14
s_4	100	0.0625
s_5	101	0.0625
s_6	110	0.0625
s_7	111	0.0625

TAB. 2 – Probabilité des différents symboles et code originaux : $l_{moyenne} = 3$

Symbole	Probabilité	Code de Huffman
s_0	0.25	00
s_1	0.21	10
s_2	0.15	010
s_3	0.14	011
s_4	0.0625	1100
s_5	0.0625	1101
s_6	0.0625	1110
s_7	0.0625	1111

TAB. 3 – Probabilité des différents symboles et code de Huffman : $l_{moyenne} = 2.79$

   - Assigner arbitrairement 0 et 1 aux deux branches conduisant au nouveau nœud formé

3. Parcourir l'arbre depuis la racine pour trouver le code affecté à chaque symbole.

Avec l'exemple du tableau 2, on obtient l'arbre présenté sur Fig. 6. On arrange les symboles par ordre décroissant. À la première étape, on regroupe les nœuds avec la plus faible probabilité ce qui correspond aux valeurs s_6 et s_7. On affecte à ce nœud la probabilité $p(s_6) + p(s_7) = 0.125$ et à chacune des branches la valeur 0 ou 1. On continue ensuite le processus, qui va regrouper s_4 et s_5. Ensuite, les probabilités restantes sont 0.25, 0.21, 0.15, 0.14 et celles des deux nouveaux nœuds : 0.125 et 0.125. Ces deux nouveaux nœuds ont encore la probabilité la plus faible, on les regroupe donc pour former un nœud de probabilité 0.25. On continue ensuite l'algorithme jusqu'à l'obtention de l'arbre complet.

Pour savoir comment sera codé un symbole, on parcours l'arbre en sens inverse (droite à gauche) pour arriver au symbole que l'on désire coder. On arrive au code final présenté dans Tab. 3.

L'entropie de ce message est égale à :

$$H(S) = -\sum_{i=0}^{7} p(s_i) \log_2(p(s_i)) = 2.781 \tag{6}$$

Au départ (Tab. 2), on avait une longueur moyenne du code qui était égale à 3. Avec le code de Huffman la longueur moyenne est passée à 2.79. Le codage de Huffman garantit une longueur moyenne comprise entre $H(S)$ et $H(S) + 1$, c'est donc un code efficace.

Le codage de Huffman a deux limitations principales.

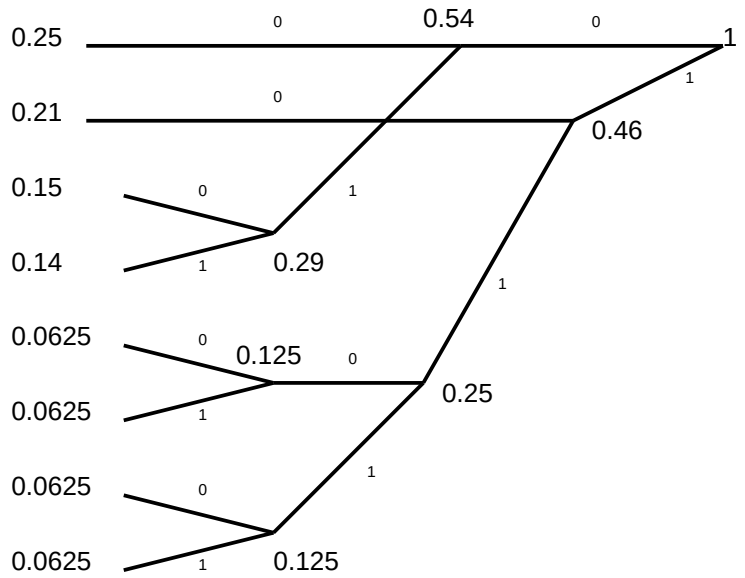


FIG. 6 – Arbre obtenu avec les symboles et les probabilités présentée dans Tab. 2

- Il n'est pas toujours facile d'évaluer les $p(s_i)$, notamment lorsque la taille des symboles augmente (pour des symboles codés sur 16 bits, il y a $2^{16} = 65536$ symboles possibles, il faut donc disposer d'une grande quantité de données pour évaluer les probabilités) ou lorsque la loi de probabilité change au cours de la transmission.
- La construction du code de Huffman fait qu'on est obligé de prendre un nombre entier de bits pour coder un symbole alors que la valeur optimale pourrait être fractionnaire.

Néanmoins, son efficacité est suffisante et sa complexité assez faible pour en faire un code largement employé. Il existe d'autres codes à longueur variable permettant de dépasser ces limites dans des cas plus spécifiques mais parfois au prix d'une complexité plus importante : Lempel-Ziv (LZ77), codes arithmétiques,...

Ce type de codage sans pertes se retrouve par exemple dans les fichiers .zip, .rar, .bz2 utilisés pour stocker des données générales.

2.5 RLE

Le RLE ou Run Length Encoding essai de tirer parti du fait que, souvent, le même symbole apparaît plusieurs fois de manière consécutive dans le message. Par exemple, la suite de symbole

AABBBBBBBBCCCC123

est clairement un codage inefficace. On peut le remplacer par

2A8B4C123

qui est beaucoup plus compact. Dans ce cas, on a le problème de l'ambigüité pour coder les chiffres, on ne saura pas si il s'agit d'un symbole ou de l'indication du nombre de répétition du symbole qui suit. Pour éviter ce problème, on utilise un caractère d'échappement (@ par exemple). Ce caractère d'échappement signale qu'un *run* va suivre, la suite indiquant la longueur (*length*) de ce run et le symbole codé par le run.

AA@8B@4C123

Pour éviter de remplacer 2 caractères par 3, on ne déclenche le run-length que lorsque 3 caractères au moins sont consécutifs.

Il est intéressant d'utiliser le RLE juste après la quantification en choisissant un ordre de parcours des coefficients permettant de regrouper les valeurs consécutives semblables. C'est d'ailleurs ce qui est fait pour la compression JPEG (cf. 4.5)

2.6 Utilisation

Ces méthodes de compression sans pertes présentent en général des performances limitées. Pour respecter les contraintes posées par le spatial, il est nécessaire de recourir à des techniques de compression avec pertes. Dans la plupart des cas, afin d'optimiser le débit au maximum, une compression sans pertes est utilisée à la suite de la compression avec pertes.

L'objectif sera de fournir au codeur sans pertes des données permettant de tirer parti au maximum de ses propriétés (concentration de l'énergie, regroupement des valeurs de même amplitude ensemble . . .).

3 Méthodes prédictives

3.1 DPCM

En regardant une image, on peut voir qu'il y a une faible variation entre des pixels voisins. On peut donc partiellement *prédire* la valeur d'un pixel en connaissant la valeur du pixel qui le précède. Une méthode simple, *Différence Pulse Code Modulation* (DPCM), consiste à considérer que le pixel sera égal à celui qui le précède, on code ensuite uniquement la valeur du premier pixel et la différence entre deux pixels consécutifs.

Les valeurs ainsi prédites seront plus proche de 0 et donc plus facile à coder par un codeur sans pertes venant ensuite.

On peut évaluer la difficulté à coder l'image obtenue en mesurant sa variance (qui correspond à la puissance du signal image). Par exemple, la variance de l'image originale (Fig. 7) est plus élevée que celle de l'erreur de prédiction (Fig. 9).

Un problème peut survenir si la transmission est altérée. Si une erreur survient sur un pixel, l'ensemble des pixels suivant sera faux. Pour éviter ce problème, on adapte le système dans le cas des transmission satellite fortement bruitées.

3.2 Les satellites SPOT 1-4

Le DPCM est utilisé pour les satellites SPOT 1 à SPOT 4. Le principal intérêt est la faible complexité du codeur. Pour éviter la propagation des erreurs, le système est modifié. Dans ce cas, le code s'appuie sur des colonnes dites pivots qui sont codées

FIG. 7 – Image originale *Barbara* ($\sigma^2 = 2227.81$)

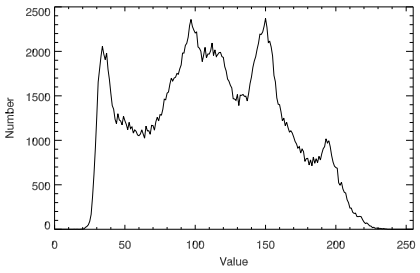


FIG. 8 – Histogramme de *Barbara*

FIG. 9 – Image de l'erreur de prédiction par DPCM pour *Barbara* ($\sigma^2 = 631.24$)

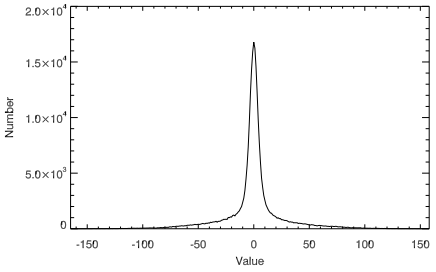


FIG. 10 – Histogramme de l'erreur de prédiction pour *Barbara*

directement sur 8 bits. Entre ces colonnes pivots, on code juste l'écart de la valeur du pixels par rapport à la moyenne des deux pixels pivots les plus proches.

On a dans ce cas :

- $s_0 = r_0$ codé sur 8 bits
- $s_1 = r_1 - \frac{1}{2}(r_0 + r_3)$ codé sur 5 bits
- $s_2 = r_2 - \frac{1}{2}(r_0 + r_3)$ codé sur 5 bits
- $s_3 = r_3$ codé sur 8 bits
- $s_4 = r_4 - \frac{1}{2}(r_3 + r_6)$ codé sur 5 bits
- ...

D'où un nombre de bits moyen de 6 et un taux de compression de 8/6=1.33. Des pertes peuvent intervenir lorsque l'erreur est trop grande et dépasse les valeurs possibles sur 5 bits. Dans ce cas, les pertes affectent un seul pixel.

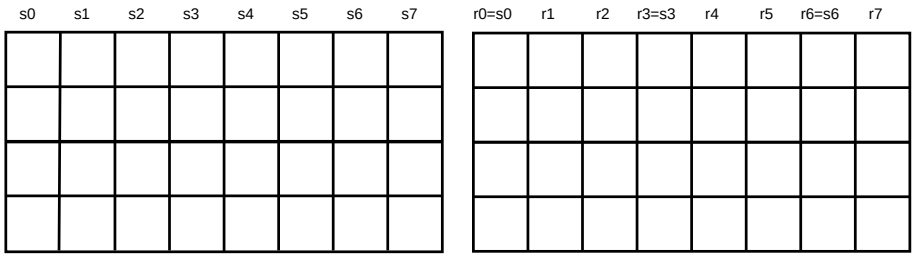


FIG. 11 – DPCM des satellites SPOT 1 à 4 sur les colonnes pivots.

4 Codage par transformée

4.1 Données

4.2 Principe du codage par transformée

Le codage par transformée est un codage très utilisé en compression d'image ou en compression vidéo. Le but de la transformée est de regrouper les informations les plus importantes pour faciliter le codage. C'est une opération mathématique qui correspond à un changement de base pour la représentation de l'image. Pour que la transformée soit efficace, il faut trouver une base sur laquelle les données seront mieux représentées.

Soit $u(m,n)$ une image de $N \times N$ pixels. On définit sa transformée $v(k,l)$ par :

$$v(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m,n) a_{k,l}(m,n) \tag{7}$$

$a_{k,l}(m,n)$ est appelée transformation. $v(k,l)$ sont les coefficients de la transformée. En général, $a_{k,l}(m,n)$ constitue une base orthonormale de l'espace image. Dans ce cas, la transformée inverse est donnée par :

$$u(m,n) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} v(k,l) a^*_{k,l}(m,n) \tag{8}$$

où $a^*_{k,l}(m,n)$ est le conjugué de $a_{k,l}(m,n)$.

Dans le cas général, le nombre d'additions et de multiplications nécessaires pour calculer les coefficients $v(k,l)$ est de l'ordre de N^4 ($O(N^4)$) ce qui est excessif pour les images de grande taille. On utilise donc généralement des transformées séparables. Une transformée est séparable dans le cas où :

$$a_{k,l}(m,n) = lin_k(m) col_l(n) \tag{9}$$

où $lin_k(m)$ est la transformée sur les lignes de l'image et $col_l(n)$ la transformée sur les colonnes de l'image. Dans ce cas, le nombre d'opérations est en général réduit à $O(N^3)$.

4.3 Utilisation de la transformée pour la compression

L'opération de transformée ne réalise pas de compression : les données ont toujours la même taille avant et après. Elle a juste réorganisé les données selon leur importance.

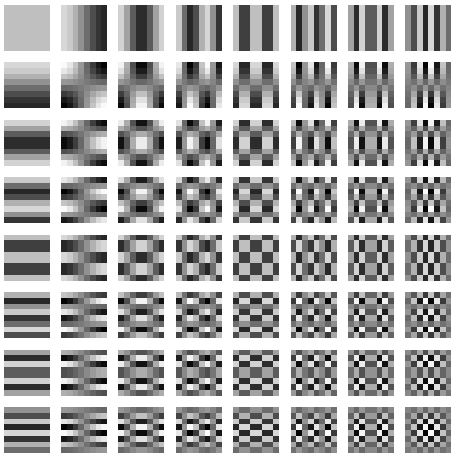


FIG. 12 – Les 64 images de base de la DCT à deux dimensions.

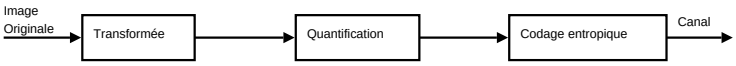


FIG. 13 – Compression par transformée. Les pertes sont dues à la quantification.

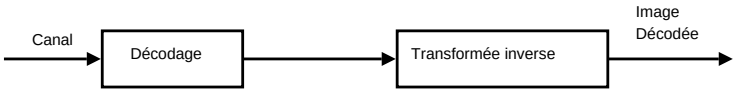


FIG. 14 – Décompression par transformée.

Pour la compression d'image, la transformée est une étape qui arrive avant la quantification et le codage (Fig. 13).

La deuxième étape est la quantification. C'est réellement durant cette étape que des données sont perdues de manière irréversible. La quantification consiste à diminuer le nombre de bits sur lequel les données sont représentées. Par exemple, sur 8 bits, on peut représenter tous les entiers de 0, 1, 2, 3, ..., 255. Si on choisit de quantifier les données sur 7 bits, on ne pourra plus représenter que 128 valeurs différentes. On choisira alors par exemple de ne représenter que les entiers pairs 0, 2, 4, ..., 254 (un pas de quantification de 2). On arrondit chacun des nombres impairs sur 8 bits à un entier pair. C'est donc cette opération d'arrondi qui est irréversible. La théorie de la quantification est un domaine riche qui ne sera pas abordé ici.

Comme cette étape intervient après la transformée, on peut choisir de mieux conserver certaines informations. Dans le cas du codage JPEG (cf. 4.5), on choisit de mieux préserver les basses fréquences en utilisant un pas de quantification plus faible pour les basses fréquences que pour les hautes fréquences.

Une fois les données transformées et quantifiées, on utilise un système de codage sans pertes. Comme les données ont été organisées par la transformée, beaucoup de données vont être quantifiées à zéro, donc le codage entropique fonctionnera bien.

4.4 Transformée DCT

Une transformée très utilisée en traitement d'image est la DCT (Discrete Cosine Transform). Elle est généralement utilisée sur des blocs de 8×8 pixels successivement sur les lignes et les colonnes (séparable). Elle est définie par (avec $N = 8$) :

$$v(k) = \alpha_k \sum_{n=0}^{N-1} u(n) \cos \frac{\pi(2n+1)k}{2N} \tag{10}$$

où

$$\alpha_k = \begin{cases} \sqrt{\frac{1}{N}} & \text{si } k = 0 \\ \sqrt{\frac{2}{N}} & \text{sinon} \end{cases} \tag{11}$$

Et sa transformée inverse par :

$$u(n) = \sum_{k=0}^{N-1} \alpha_k v(k) \cos \frac{\pi(2n+1)k}{2N} \tag{12}$$

L'avantages de la DCT est que c'est une transformation en général très proche de la transformée optimale (qui est la KLT ou Karhunen-Loeve Transform). La KLT qui est optimale est constituée des vecteurs propres de l'image, elle dépend donc de chaque image et la matrice de transformation doit être recalculée pour chaque image. Son utilisation est donc délicate. La DCT est indépendante de l'image et a des performances très proches de la KLT c'est donc la DCT qui est utilisée dans de nombreuses applications. La DCT étant une opération mathématique proche de la transformée de Fourier, des algorithmes rapides existent pour la calculer (sur le principe de la FFT ou Fast Fourier Transform).

112	77	56	66	157	204	204	198
108	74	53	87	177	207	204	204
72	57	66	126	197	209	206	208
55	65	113	173	207	211	210	211
77	107	160	198	208	206	208	207
127	158	188	202	203	202	202	206
163	186	197	198	202	205	207	209
182	187	193	197	199	203	206	209

1322	-311	-48	52	14	-13	2	7
-192	-145	45	96	17	-13	3	9
9	64	92	45	-13	-12	3	4
14	31	6	-17	-20	-2	4	2
-5	-12	-13	-10	-3	9	4	1
-1	-6	-3	-3	-3	0	1	0
-6	0	-1	0	2	0	0	-1
-2	-3	-2	0	-1	-1	-1	1

FIG. 15 – Valeurs du détail de l'image *Lena* (Fig. 2) et la transformée par DCT. On voit que l'énergie est concentrée dans les coefficients basses fréquences (en haut à gauche)

4.5 Compression JPEG

Le JPEG est un standard de compression défini par le CCITT (Consultative Committee for International Telephone and Telegraph, actuellement ITU-T) et l'ISO (International Standards Organization). Le travail a commencé en juin 1987 pour aboutir à une première proposition en 1991. JPEG est un acronyme pour Joint Photographic Experts Group. L'intervention de nombreux partenaires dans sa définition en a fait un standard largement utilisé (images sur internet, standard pour la plupart des appareils photo numériques, ...). Son principe a également inspiré les algorithmes de compression vidéo (MPEG-2 utilisé pour les DVD).

La compression JPEG est basée sur une transformée par DCT sur les blocs 8×8 de l'image. La compression peut se diviser en différentes étapes comme le montre Fig. 16.

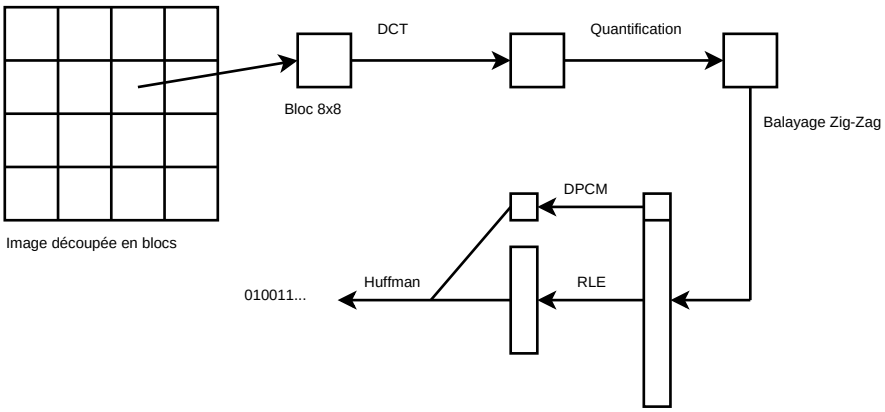


FIG. 16 – Étapes du codage JPEG

- L'image est divisée en bloc de 8×8 pixels (Fig. 19).
- Une DCT est appliquée successivement sur les lignes et les colonnes de l'image (séparable). Cette transformée permet de concentrer l'énergie sur quelques coefficients (Fig. 20).
- Les coefficients de la DCT sont quantifiés. C'est cette étape qui rend la compression irréversible (on peut voir la différence entre Fig. 19 et Fig. 22). Différentes tables de quantification peuvent être utilisées selon les objectifs de compression (par exemple Fig. 17). Une grande partie des coefficients haute fréquence se retrouve alors réduite à zéro (Fig. 21).
- Un codage zigzag est appliqué à l'image pour ordonner les 64 coefficients. Comme les basses fréquences (horizontales et verticales) ont des coefficient plus importants, on essaie de les regrouper (Fig. 23).
- Le coefficient basse fréquence (première ligne, première colonne) est codé par DPCM par rapport au coefficient basse fréquence du bloc précédent. Les autres coefficients sont codés par RLE (Fig. 24).
- Un codage de Huffman est appliqué ensuite pour réduire au maximum la longueur du bloc.

1	3	5	7	9	11	13	15
3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29

FIG. 17 – Exemple de table de quantification

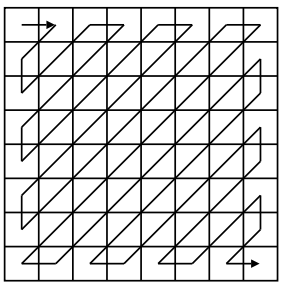


FIG. 18 – Parcours *zigzag* des coefficients d'un bloc.

112	77	56	66	157	204	204	198
108	74	53	87	177	207	204	204
72	57	66	126	197	209	206	208
55	65	113	173	207	211	210	211
77	107	160	198	208	206	208	207
127	158	188	202	203	202	202	206
163	186	197	198	202	205	207	209
182	187	193	197	199	203	206	209

FIG. 19 – Bloc original

1322	-311	-48	52	14	-13	2	7
-192	-145	45	96	17	-13	3	9
9	64	92	45	-13	-12	3	4
14	31	6	-17	-20	-2	4	2
-5	-12	-13	-10	-3	9	4	1
-1	-6	-3	-3	-3	0	1	0
-6	0	-1	0	2	0	0	-1
-2	-3	-2	0	-1	-1	-1	1

FIG. 20 – DCT du bloc original

1322	-312	-50	49	18	-11	0	0
-192	-145	42	99	22	-13	0	17
10	63	90	44	-13	-15	0	0
14	27	11	-13	-15	0	0	0
-9	-11	-13	-15	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

FIG. 21 – DCT quantifiée

113	80	49	69	162	200	203	200
104	69	51	90	178	207	206	202
76	57	71	129	198	211	207	206
54	65	113	173	208	207	206	210
75	106	157	199	206	204	206	210
129	157	186	205	200	206	207	208
166	186	194	202	196	207	205	209
174	192	194	201	195	205	202	212

FIG. 22 – DCT inverse

Les performances du codage JPEG sont assez bonnes et souvent on ne peut pas voir les dégradations pour des taux inférieurs à 10 :1 ou 20 :1. À faible débit (fort taux de compression), l'apparition d'un effet de blocs peut être gênante.

4.6 Adaptation SPOT-5

Le système de compression du satellite d'observation SPOT-5 est très semblable à la compression JPEG. Il est nécessaire de faire quelques modifications pour s'adapter aux contraintes spatiales. La régulation de débit notamment est une étape cruciale. L'adaptation se fait en modifiant la table de quantification des coefficients.

5 Ondelettes et codage progressif

5.1 Intérêt du codage progressif

Les méthodes présentées précédemment sont efficaces mais présentent un inconvénient. Il est possible d'ajuster le débit en changeant la table de quantification, mais le débit visé doit être décidé à la compression. Si les mêmes données sont destinées à plusieurs usages, chacun nécessitant un débit différent, on devra garder autant de versions compressées des données que d'usages prévus. D'autre part, si la transmission d'un message est interrompue, les données reçues seront inutilisables sans l'ensemble du message.

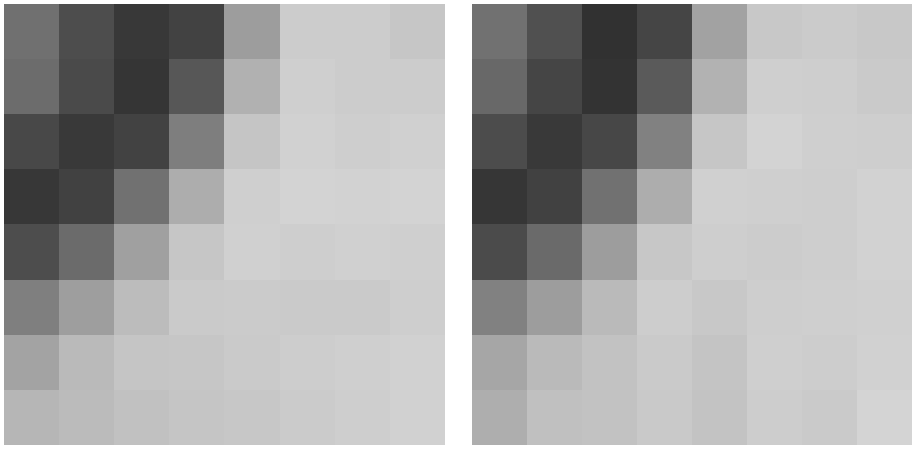
Le codage progressif permet d'éviter ces inconvénients. L'idée est que le message codé peut être tronqué à n'importe quel point, donnant un message décodable, présentant une distortion plus forte, mais un débit plus faible. Les ondelettes et les systèmes de codage par plan de bits ont permis la définition de ces algorithmes.

1322, -312, -192, 10, -145, -50, 49, 42, 63, 14, -9, 27, 90, 99, 18, -11, 22, 44, 11, -11, 0, 0, 0, -13, -13, -13, -13, 0, 0, 0, -15, -15, -15, 0, 0, 0, 0, 0, 0, 0, 0, 0, 17, 0, 0, 0, 0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

FIG. 23 – Parcours des coefficients selon Fig. 18

1322, -312, -192, 10, -145, -50, 49, 42, 63, 14, -9, 27, 90, 99, 18, -11, 22, 44, 11, -11, @, 3, 0 , @, 4, -13 , @, 3, 0, @, 3, -15 , @, 9, 0 , 17, @, 21, 0

FIG. 24 – Codage RLE



(a) (b)

FIG. 25 – Comparaison entre le bloc original (a) et l'erreur commise par quantification de la DCT (b). L'erreur commise est faible malgré une quantification importante des hautes fréquences.

5.2 Ondelettes

La transformée en ondelettes ne nécessite pas de partager l'image en blocs 8×8, il n'y a donc pas d'effets de blocs apparaissant comme dans le cas de JPEG. A faible débit, la transformée en ondelettes est donc beaucoup plus intéressante. Il existe aussi des structures pour effectuer les calculs de manière rapide (structure de lifting scheme).

L'image est filtrée par des filtres passe-haut et passe-bas, successivement selon les directions horizontales et verticales (ondelettes séparables). En général, le filtrage est itéré sur la bande passe-bas donnant une structure similaire à Fig. 26.

La structure de la décomposition de la transformée en ondelettes se prête bien à un codage progressif. On peut commencer par transmettre la sous-bande passe-bas puis raffiner progressivement avec les détails passe-haut.

5.3 La norme JPEG 2000

Le travail sur la norme JPEG 2000 a commencé peu après la publication de la norme JPEG. Ce travail a été motivé principalement pour apporter plus de souplesse ainsi que des taux de compression plus importants. Il y a notamment moins de limitations concernant la taille des images, le nombre de bandes, la dynamique des pixels [6].

Le standard JPEG 2000 est basé sur une transformée en ondelettes. Selon le mode

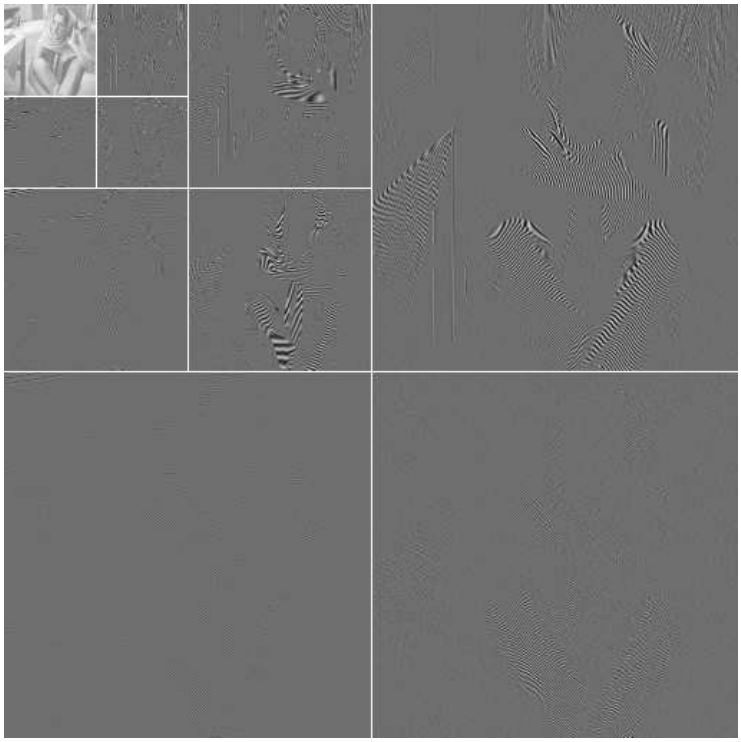


FIG. 26 – Décomposition multirésolution de *Barbara*

de compression choisi (avec ou sans pertes), différentes ondelettes sont utilisées. Dans le cas de la compression avec pertes, c'est une ondelette biorthogonale à support compact qui est utilisée. Souvent appelée (9,7) (filtre passe-bas longueur 9 et filtre passe-haut de longueur 7), il s'agit de fonctions à 4 moments nuls. Les détails théoriques concernant cette ondelette peuvent être trouvés dans [4].

Une grande partie de l'efficacité de JPEG 2000 est aussi à attribuer au codeur entropique utilisé EBCOT (Embedded Block Coding) permettant le codage progressif.

5.4 Utilisation pour Pléiades

C'est un système similaire qui est prévu pour le satellite Pléiades. La transformée en ondelettes est appliquée sur des blocs de l'image. En effet, il est nécessaire d'effectuer la transformée au fur et à mesure que les données arrivent (compression *au fil de l'eau*) car le satellite peut acquérir des données en continu.

Des blocs d'environ 4000 pixels sont donc compressés par un système similaire à JPEG 2000. Le problème reste encore ici la régulation de débit, il faut s'assurer que le niveau des buffers (mémoire) de compression reste constant malgré la variabilité des scènes observées.

6 Conclusion

Les techniques de compression utilisées pour les images satellites sont semblables à ce qui est fait dans les autres domaines. L'amélioration des instruments : meilleure résolution spatiale (plus de pixels), meilleure résolution radiométrique (plus de bits pour coder chaque pixel), meilleure résolution spectrale (plus de bande pour chaque image) conduit à une augmentation de la quantité de données. Toutefois, les applications scientifiques de pointe pour lesquelles les images satellites sont souvent acquises nécessitent des données de grande qualité. La compression qui est devenue une étape indispensable au cours des dernières années doit donc s'attacher particulièrement à préserver la qualité des données.

Références

[1] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*, U. of Illinois, Ed. Illini Books, 1949.

[2] A. K. Jain, *Fundamentals of digital image processing*, N. Englewood Cliffs, Ed. Prentice-Hall, 1989.

[3] M. Rabbani and P. W. Jones, *Digital Image Compression Techniques*. SPIE - The International Society for Optical Engineering, 1991.

[4] S. Mallat, *A Wavelet Tour of Signal Processing*. Academic Press, 1997.

[5] D. Salomon, *Data Compression*, second edition ed. Springer-Verlag New York, Inc, 2002.

[6] D. S. Taubman and M. W. Marcelin, *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers, 2002.